

Express Mailing Label No. EL487542374US

PATENT APPLICATION
Docket No. 3000.2.27
IBM No. ST9-99-151

UNITED STATES PATENT APPLICATION

of

HONGHAI SHEN

and

YUDONG SUN

for

PROVIDING DYNAMIC WEB PAGES BY SEPARATING
SCRIPTS AND HTML CODE

MADSON & METCALF, P.C.

ATTORNEYS AT LAW
900 GATEWAY TOWER WEST
15 WEST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84101

PROVIDING DYNAMIC WEB PAGES BY SEPARATING SCRIPTS AND HTML CODE

BACKGROUND OF THE INVENTION

Field of the Invention

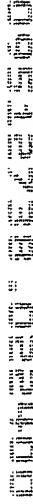
The present invention relates generally to programming techniques in the hypertext markup language ("HTML"). More particularly, the present invention relates to a system and method for providing dynamic Web pages by separating scripts and HTML code.

Identification of Copyright

A portion of the disclosure of this patent document contains material subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

Relevant Technology

The World Wide Web (hereinafter "the Web") is a collection of Internet-accessible servers from which specially formatted documents may be retrieved and displayed by Web browsers, such as Netscape Navigator™ and Microsoft Internet Explorer™. Currently, the hypertext markup language ("HTML") is the most



1 common authoring language for creating Web documents, also known as "Web
2 pages." A Web page is identified by a uniform resource locator ("URL"), which is
3 used by a Web browser to locate and display a particular Web page.

4 Until recently, most Web pages were static, i.e. the content of a Web page
5 was the same each time it was displayed. As a result, to produce customized
6 content for different users, for example, different Web pages (with different URLs)
7 were created in advance for each user. Such an approach has obvious shortcomings,
8 however, since a Web server would need to store millions of different Web pages
9 for millions of different users.

10 Consequently, techniques were developed to make Web pages dynamic, i.e.
11 the content of a single Web page may change each time it is viewed. A different
12 Web page may be displayed depending, for example, on the identity of the reader,
13 the geography of the reader, the time of day, previous pages viewed by the reader,
14 and the like.

15 For instance, a user may retrieve a Web page containing her bank account
16 balance. However, the bank does not typically store individual "account balance"
17 pages for each user. Instead, user-specific information is retrieved from the bank's
18 database and dynamically inserted into a Web page template, after which the
19 resulting Web page is sent to the user's Web browser.

20 Conventionally, dynamic Web pages are created by embedding server-side
21 scripts in Web pages, which execute on a Web server and generate HTML elements

1 prior to the Web page being sent to a browser. A variety of technologies exist for
2 producing dynamic HTML pages, including common gateway interface ("CGI")
3 scripts, active server pages ("ASP"), server-side includes ("SSI"), cookies, Java,
4 JavaScript, and ActiveX.

5 Unfortunately, embedding server-side scripts within Web pages has at least
6 two major drawbacks that have plagued Web page developers and HTML
7 programmers. First, a Web page including embedded scripts cannot be effectively
8 edited with an interactive HTML editor, because some of the HTML elements of the
9 page are only generated by the scripts at run time, and are thus unknown to the
10 editor at design time.

11 Second, Web documents including embedded scripts are often difficult to
12 maintain and debug since the scripts are typically scattered throughout a Web page
13 at various locations at which corresponding HTML elements are to be inserted by
14 a Web server. This fact also makes it difficult to provide a high-level integrated
15 development environment ("IDE") for script writers and programmers.

16 The above-described problems are more fully illustrated by the following
17 example. A user may wish to update her personal information on an e-commerce
18 site, such as Amazon.com™. Accordingly, she may request a dynamic Web page
19 adapted for that purpose by clicking on a corresponding button displayed by her
20 Web browser.

1 Figure 1 depicts a conventional dynamic Web page 2 for updating a user's
2 personal information. As illustrated, the Web page 2 includes a number of
3 embedded scripts 4. The Web page 2 may be an active server page ("ASP"), as
4 shown, although other technologies could be used.

5 In general, the embedded scripts 4 are unintelligible to Web browsers and
6 HTML editors. As such, if the Web page 2 of Figure 1 is displayed by a standard
7 Web browser or HTML editor, the scripts will be ignored, and a displayed page 2
8 similar to that of Figure 2 will result.

9 *Sub A* Conventionally, the Web server handing the request modifies the Web page
10 2 by replacing the scripts 4 of Figure 1 with the output of the script execution (e.g.,
11 the Write() arguments). Typically, the output includes one or more HTML elements
12 6, as illustrated in Figure 3. Thereafter, a “modified” Web page 8 may be sent to the
13 requesting Web browser.

14 For instance, the Web server may replace the script 4A of Figure 1, i.e. <%
15 Response.Write("<input name='name' value=' " & rs("name") &
16 " ' >") %>, with the HTML element 6A of Figure 3, i.e. <input name='name'
17 value='Jane Doe'>. The rs("name") argument is a database query that
18 returns, for example, the user's name, i.e. "Jane Doe."

19 The modified Web page 8, as displayed by a Web browser, is shown in
20 Figure 4. As a result of the above-described process, a single requested Web page

1 2 may produce customized output for different users. In other words, the Web page
2 2 is "dynamic."

3 Unfortunately, conventional dynamic Web pages 2 of the type illustrated in
4 Figure 1 have numerous drawbacks. As previously noted, a Web page 2 including
5 embedded scripts 4 cannot be effectively edited by an HTML editor, because some
6 HTML elements 6 do not exist until after the scripts 4 are executed, and are thus not
7 available to the HTML editor at design time.

8 For example, the HTML element 6A, i.e. `<input name='name'
9 value='Jane Doe'>`, does not exist in the Web page 2 of Figure 1. The element
10 6A is not added until after the Web server executes the script 4A. As a result, a
11 Web designer is limited to the displaying and editing the incomplete Web page 2
12 of Figure 2, rather than the completed Web page 2 of Figure 4. Designing and
13 laying out a Web page 2 is understandably difficult when some of the HTML
14 elements 6 are not available at design time.

15 Moreover, debugging and maintaining conventional dynamic Web pages
16 2 are difficult, since individual scripts 4 are scattered throughout the pages 2 at
17 various locations dictated by the insertion points of corresponding HTML elements
18 6. For instance, the Web page 2 of Figure 1 includes three different scripts 4A-C at
19 three different locations. Larger Web pages 2 may include hundreds of scripts. The
20 lack of a single location in which a Web designer may look to find all of the
21 embedded scripts 4 is a serious problem in Web page development.

1 Accordingly, what is needed is a system and method for providing dynamic
2 Web pages. What is also needed is a system and method for providing dynamic
3 Web pages that may be edited by an interactive HTML editor. What is also needed
4 is a system and method for providing dynamic Web pages by separating scripts and
5 HTML code.

6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

1

SUMMARY OF THE INVENTION

2 The present invention solves many or all of the foregoing problems by
3 introducing a novel system and method for providing dynamic Web pages by
4 separating scripts and HTML code.

5 *S. J. M.* In one aspect of the invention, a request reception module may receive a
6 request for a document stored within document server. The document may be
7 encoded in the hypertext markup language (HTML) and may include one or more
8 HTML elements.

9 After the request is received, a parsing module may parse the requested
10 document to generate therefrom a corresponding document object model (DOM)
11 including at least one object. Each HTML element of the document typically
12 corresponds to one DOM object.

13 After the document is parsed, an instruction obtaining module may obtain
14 a transformation instruction directed to at least one object of the DOM. A variety
15 of transformation instructions are possible. For example, one transformation
16 instruction may retrieve a value from a database and assign the value to a DOM
17 object. Another transformation instruction may replace one object with a different
18 object.

19 In another aspect of the invention, the instruction obtaining module may
20 include a script file access module, which may read a transformation instruction
21 from a script file corresponding to the requested document. In one embodiment,

1 the script file, including one or more transformation instructions, may be included
2 within a separate portion of the document. For example, the HTML elements of the
3 document and the transformation instructions of the script file may be stored within
4 separate portions of a single logical data file. In another embodiment, however, the
5 script file and the document may comprise logically separate data files.

6 After the transformation instruction is obtained, an object transformation
7 module may transform the first object in accordance with the transformation
8 instruction. Thereafter, a flattening module may flatten the DOM to generate
9 therefrom a corresponding transformed document.

10 In one embodiment, the transformed document may comprise one or more
11 HTML elements corresponding to the objects of the DOM. As a result of the
12 flattening process, any transformation of a DOM object is preferably reflected within
13 a corresponding HTML element of the transformed document.

14 In yet another aspect of the invention, a transmission module may transmit
15 the transformed document to a requesting client program. In various embodiments,
16 the client program may include a standard Web browser.

17 These and other objects, features, and advantages of the present invention
18 will become more fully apparent from the following description and appended
19 claims, or may be learned by the practice of the invention as set forth hereinafter.

20
21

1

BRIEF DESCRIPTION OF THE DRAWINGS

2 The present invention is more fully disclosed in the following specification,
3 with reference to the accompanying drawings, in which:

4 Figure 1 is an illustration of a Web document;

5 Figure 2 is an illustration of a screen display produced by a Web browser;

6 Figure 3 is an illustration of a Web document;

7 Figure 4 is an illustration of a screen display produced by a Web browser;

8 Figure 5 is a schematic block diagram of a computer system suitable for
9 hosting a plurality of software modules according to an embodiment of the
10 invention;

11 Figure 6 is a schematic block diagram of a system for providing dynamic
12 Web pages according to an embodiment of the invention;

13 Figure 7 is a schematic flowchart of a method for providing dynamic Web
14 pages according to an embodiment of the invention;

15 Figure 8 is an illustration of a Web document according to an embodiment
16 of the invention;

17 Figure 9 is an illustration of a screen display generated by a Web browser
18 according to an embodiment of the invention;

19 Figure 10 is an illustration of a Document Object Model (DOM) according
20 to an embodiment of the invention;

21

1 Figure 11 is an illustration of a script file according to an embodiment of the
2 invention;

3 Figure 12 is an illustration of a DOM according to an embodiment of the
4 invention;

5 Figure 12 is an illustration of a DOM according to an embodiment of the
6 invention;

7 Figure 14 is an illustration of a Web document according to an embodiment
8 of the invention; and

9 Figure 15 is an illustration of a screen display generated by a Web browser
10 according to an embodiment of the invention.

11

12

13

14

15

16

17

18

19

20

21

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Certain presently preferred embodiments of the invention are now described with reference to the Figures, where like reference numbers indicate identical or functionally similar elements. The components of the present invention, as generally described and illustrated in the Figures, may be implemented in a variety of configurations. Thus, the following more detailed description of the embodiments of the system and method of the present invention, as represented in the Figures, is not intended to limit the scope of the invention, as claimed, but is merely representative of presently preferred embodiments of the invention.

Throughout the following description, various system components are referred to as "modules." In certain embodiments, the modules may be implemented as software, hardware, firmware, or any combination thereof.

For example, as used herein, a module may include any type of computer instruction or computer executable code located within a memory device and/or transmitted as electronic signals over a system bus or network. An identified module may include, for instance, one or more physical or logical blocks of computer instructions, which may be organized as an object, a procedure, a function, or the like.

The identified modules need not be located physically together, but may include disparate instructions stored at different memory locations, which together implement the described logical functionality of the module. Indeed, a module may

1 include a single instruction, or many instructions, and may even be distributed
2 among several discrete code segments, within different programs, and across
3 several memory devices.

4 Figure 5 is a schematic block diagram illustrating a computer system 10 in
5 which a plurality of software modules may be hosted on one or more computer
6 workstations 12 connected via a network 14. The network 14 may include a wide
7 area network (WAN) or local area network (LAN) and may also include an
8 interconnected system of networks, one particular example of which is the Internet.

9 A typical computer workstation 12 may include a central processing unit
10 (CPU) 16. The CPU 16 may be operably connected to one or more memory devices
11 18. The memory devices 18 are depicted as including a non-volatile storage device
12 20 (such as a hard disk drive or CD-ROM drive), a read-only memory (ROM) 22,
13 and a random access memory (RAM) 24.

14 The computer workstation 12 may operate under the control of an operating
15 system (OS) 25, such as OS/2®, WINDOWS NT®, WINDOWS®, UNIX®, and the like.
16 In various embodiments, the OS 25 provides a graphical user interface (GUI).

17 The computer workstation 12 may also include one or more input devices
18 26, such as a mouse and/or a keyboard, for receiving inputs from a user. Similarly,
19 one or more output devices 28, such as a monitor and/or a printer, may be provided
20 within, or be accessible from, the computer workstation 12.

1 A network interface 30, such as an Ethernet adapter, may be provided for
2 coupling the computer workstation 12 to the network 14. Where the network 14 is
3 remote from the computer workstation 12, the network interface 30 may include a
4 modem, and may connect to the network 14 through a local access line, such as a
5 telephone line.

6 Within any given computer workstation 12, a system bus 32 may operably
7 interconnect the CPU 16, the memory devices 18, the input devices 26, the output
8 devices 28, the network interface 30, and one or more additional ports 34, such as
9 parallel and/or serial ports.

10 The system bus 32 and a network backbone 36 may be regarded as data
11 carriers. Accordingly, the system bus 32 and the network backbone 36 may be
12 embodied in numerous configurations, such as wire and/or fiber optic lines, as well
13 as electromagnetic communication channels using visible light, infrared, and radio
14 frequencies.

15 The computer workstations 12 may be coupled via the network 14 to
16 application servers 42, and/or other resources or peripherals 44, such as scanners,
17 fax machines, and the like. External networks, such as the Internet 40, may be
18 coupled to the network 14 through a router 38 or firewall.

19 In various embodiments, one or more Web servers 46 may be accessible to
20 the workstations 12 via the Internet 40. A Web server 46 is a computer system, such
21 as a workstation 12, including specialized software for delivering (serving) Web

100-1000-0000-0000-0000

1 pages to Web browsers. A variety of Web server application programs are
2 available, including public domain software from NCSA and Apache, as well as
3 commercial packages from Microsoft, Netscape and others.

4 Referring now to Figure 6, a system 50 for providing dynamic Web pages
5 according to a presently preferred embodiment of invention may include a
6 workstation 12 and a Web server 46. The workstation 12 may include a conventional
7 Web browser 52, such as Netscape Navigator™ or Microsoft Internet Explorer™,
8 which is capable of communicating with the Web server 46 using the hypertext
9 transfer protocol ("HTTP").

10 The Web server 46 is depicted as including a request reception module 54.
11 In one embodiment, the request reception module 54 receives (from the Web
12 browser 52) a request for a document 56 stored within a document storage area 58
13 of the Web server 46. The document 56 may be encoded in the hypertext markup
14 language ("HTML") and may include one or more HTML elements 6, as described
15 more fully hereafter.

16 In one embodiment, the Web server 46 also includes a parsing module 60,
17 commonly referred to as a "parser." The parsing module 60 retrieves, in various
18 embodiments, the requested document 56 and parses the document 56 to generate
19 therefrom a corresponding Document Object Model (DOM) 62, sometimes referred
20 to as a "parse tree." A DOM 62 is a tree-like, hierarchical data structure including
21

1 one or more objects 64 that represent the various HTML elements 6 of the document
2 56.

3 In certain embodiments, the parsing module 60 is a conventional HTML
4 parser. For example, both Netscape Navigator™ and Microsoft Internet Explorer™
5 include HTML parsers, which may be adapted, in various embodiments, for use
6 within the Web server 46. In an alternative embodiment, a custom HTML parser
7 may be used. Conventionally, however, a Web server 46 does not include a parsing
8 module 60, since a document 56 is normally parsed only by a Web browser 52 at the
9 time the document 56 is displayed.

10 The Web server 46 may also include a script file access module 66. In certain
11 embodiments, the script file access module 66 may be configured to retrieve a script
12 file 68 (from a script file storage area 70) corresponding to the requested document
13 56, as explained more fully hereafter.

14 A script file 68 may contain one or more transformation instructions 72 or
15 "scripts." However, unlike the conventional server-side scripts 4 of Figure 1, each
16 transformation instruction 72 is directed to at least one object 64 of the DOM 62 and
17 includes at least one transformation to be performed on the at least one object 64.

18 For example, as described in greater detail below, one transformation
19 instruction 72 may query a database for a value and assign the value to an object 64
20 of the DOM 62. Another transformation instruction 72 may replace one object 64
21 with a different object 64. A wide variety of transformation instructions 72 are

1 possible within the scope of the invention. A transformation instruction 72 may
2 have any suitable syntax, so long as it identifies at least one object 64 and at least
3 one transformation.

4 In the depicted embodiment, the script files 68 and the Web documents 56
5 comprise logically separate data files, and may even be housed within separate
6 storage areas 58, 70 of the Web server 46. In such an embodiment, a document 56
7 and a corresponding script file 68 may have identical or similar names, with the
8 exception of a file extension or other delimiter. For example, if the requested
9 document 56 is named "personalinfo.html," the corresponding script file 68 may be
10 named "personalinfo.scr" or the like. Thus, in various embodiments, the request
11 reception module 54 may identify a corresponding script file 68 for any requested
12 document 56.

13 In an alternative embodiment, the Web browser 52 may request a script file
14 68, and the request reception module 54 may identify a corresponding document 56
15 in like manner. In one embodiment, the script file 68, including one or more
16 transformation instructions 72, may be included within a separate portion of the
17 document 56. For example, the HTML elements 6 of the document 56 and the
18 transformation instructions 72 of the script file 68 may be stored within separate
19 portions of a single logical data file.

20 In certain presently preferred embodiments, the Web server 46 also includes
21 a object transformation module 74, which may transform one or more objects 64 of

1 the DOM 62 in accordance with the transformation instruction(s) 72 of a
2 corresponding script file 68. In various embodiments, the object transformation
3 module 74 may retrieve each instruction 72 from the script file 68, in sequence, and
4 performs the requested transformation(s).

5 In the depicted embodiment, the object transformation module 74 includes
6 a number of supplemental modules for performing various transformation
7 instructions 74. For example, a database query module 76 may be provided for
8 performing a specified query on a database 78 to retrieve a value. Likewise, a value
9 assignment module 80 may be provided for assigning a value to a DOM object 64.
10 Moreover, an object replacement module 82 may be provided to replace one object
11 64 of the DOM 62 with another object 64.

12 The Web server 46 may also include a flattening module 84. In various
13 embodiments, the flattening module 84 flattens the DOM 62 to generate therefrom
14 a transformed document 86. As used herein, the term "flattening" refers to a
15 process of converting the DOM 62 back into an equivalent HTML document 86
16 including one or more corresponding HTML elements 6. Techniques for flattening
17 a DOM 62 are well known in the art. The resulting document 86 is designated
18 as "transformed" because any transformations of the DOM objects 64 are preferably
19 reflected in the corresponding HTML elements 6 of the transformed document 86.

20 In various embodiments, the Web server 46 may also include a transmission
21 module 88. The transmission module 88 may send the transformed document 86

1 (via the Internet 40) to the Workstation 12, such that the document 86 may be
2 displayed by the Web browser 52.

3 Referring now to Figure 7, a schematic flowchart includes a method 100 for
4 providing dynamic Web pages according to a presently preferred embodiment of
5 the invention. The method 100 may begin by receiving 102, at a Web server 46, a
6 request for a document 56.

7 Figure 8 illustrates an exemplary document 56 according to an embodiment
8 of the invention. For purposes of comparison, the document 56 may be configured,
9 like the Web page 2 of Figure 1, to update a user's personal information on an e-
10 commerce site, such as Amazon.com™. However, unlike the Web page 2 of Figure
11 1, the document 56 need not include conventional embedded scripts 4.

12 For example, rather than including a script 4A, as in Figure 1, a regular
13 HTML element 6D, i.e. `<input name='name' value = ''>`, may be used. As
14 illustrated, the element 6D may be similar to the element 6A of Figure 3 (which was
15 inserted by the Web server after executing the script 4A), except that the "value"
16 attribute may be left empty.

17 If displayed by a Web browser, the document 56 may appear as shown in
18 Figure 9. Visually, the displayed document 56 is very similar to that of Figure 4,
19 with the exception of the customized personal data. Thus, unlike the Web page 2
20 of Figure 1, the document 56 of Figure 8 may be effectively edited by an HTML
21 editor, because all of the necessary HTML elements 6 are included. This is a great

1 advantage to Web designers and HTML programmers, who need to edit a
2 document that is as similar to the desired end product (i.e. Figure 4) as possible.

3 After the document request is received 102, the method 100 may continue
4 by parsing 104 the document 56 to generate therefrom a corresponding Document
5 Object Model (DOM) 62. As noted, a DOM 62 is a tree-like, hierarchical data
6 structure including one or more objects 64 that represent the HTML elements 6 of
7 the document 56. Figure 10 illustrates a portion of a simplified DOM 62
8 corresponding to the document 56 of Figure 8.

9 After the document 56 is parsed 104, the method 100 may continue by
10 identifying 106 a script file 68 corresponding to the document 56. In certain
11 embodiments, a document 56 and a corresponding script file 68 may have identical
12 or similar names, with the exception of a file extension or other delimiter. For
13 example, if the requested document 56 is named “personalinfo.html,” the
14 corresponding script file 68 may be named “personalinfo.scr” or the like.

15 Figure 11 illustrates an exemplary script file 68 in accordance with an
16 embodiment of the invention. As previously noted, a script file 68 may include one
17 or more transformation instructions 72. Each transformation instruction may be
18 directed to at least one object 64 of the DOM 62 and indicate at least one
19 transformation to be performed on the at least one object 64.

20 While the script file 68 and the document 56 are depicted herein as logically
21 separate data files, the script file 68 may be included, in some instances, within a

1 separate portion of document 56. For example, all of the transformation instructions
2 72 of the script file 68 may be located, as a group, at the beginning of the document
3 56:

```
4      <%
5          dom.allElements("name").value = rs("name")
6          dom.allElements("phone").value = rs("phone")
7          dom.allElements("email").value = rs("email")
8      %>
9      <html>
10     <head>
11         <title>Personal Information Update</title>
12     </head>
13     <body>
14         <table>
15             .
16             .
17             .
18         </table>
19     </body>
20 </html>
```

10 The location of the transformation instructions 72 within the document 56 is not
11 crucial. However, all of the instructions 72 should be located together to avoid the
12 noted drawbacks of conventional dynamic Web pages 2.

13 After the script file 68 is identified 106, the method 100 may continue by
14 reading 108 a transformation instruction 72 from the script file 68. Thereafter, the
15 method 100 may continue by transforming 110 one or more objects 64 of the DOM
16 62 in accordance with the read transformation instruction 72.

17 For example, if the first transformation instruction 72A is read, i.e.
18 dom.allElements("name").value = rs("name"), the method 100 may
19 proceed to transform the object 64A of Figure 10 by querying a database 78 for a
20 value, i.e. the user's name, and assigning the value to the object 64A. As previously
21

1 noted, the "rs" (recordset) argument indicates a database query in one embodiment.
2 If, for instance, the user's name is "Jane Doe," Figure 12 illustrates the transformed
3 object 64A within the DOM 62.

4 One advantage of transforming a DOM 62 rather than modifying an Web
5 document 56, itself, as in conventional approaches, is that the DOM 62 is more easily
6 transformed than HTML text. For example, any object 64 of the DOM 62 may be
7 randomly accessed and transformed by a simple command, whereas modifying a
8 Web document 56 requires more complex manipulations of the HTML text, such as
9 cutting and pasting HTML elements 6.

10 After the transformation step 110 is complete, the method 100 may continue
11 by determining 112 whether the script file 68 includes more transformation
12 instructions 112. If so, the method may returns to step 108 to read the next
13 instruction 72.

14 If, however, all of the instructions 72 have been used, the method 100
15 continues by flattening 114 the DOM 62 to create a transformed document 86. As
16 previously noted, the flattening process involves converting the DOM 62 back into
17 an HTML document 86. Consequently, any transformations to the DOM objects 64
18 will be preferably reflected in the corresponding HTML elements 6 of the document
19 86.

20 For example, Figure 13 illustrates the DOM 62 after execution of the three
21 transformations instructions 72A-C. After the flattening step 114, the transformed

1 document 86 of Figure 14 may result, which may then be sent 116 to the requesting
2 Web browser 52 and displayed, as illustrated in Figure 15.

3 Surprisingly, the transformed document 86 is identical to the modified
4 document 8 of Figure 3, which was produced by conventional techniques using
5 embedded server-side scripts 4. However, the transformed document 86, in
6 accordance with an embodiment of the invention, does not rely on embedded
7 scripts 4. Rather, the transformed document 86 is generated, as noted above, by
8 parsing a requested document 56 at the Web server 46, transforming the resulting
9 DOM 62, and flattening the DOM 62 into a “transformed” HTML document 86.

10 Moreover, unlike conventional approaches, the transformation instructions
11 72 are not “place holders” for HTML elements 6 to be inserted later by a Web server
12 46. As such, all of the transformations instructions 72 can be located together, even
13 within a separate script file 68.

14 Importantly, a document 56 in accordance with the present invention may
15 be effectively edited by an HTML editor, since all of the HTML elements may be
16 included within the document 56 at design time. Any transformations, such as
17 assignments of values and the like, may be accomplished by transforming the DOM
18 62. No embedded scripts 4 are necessary.

19 The present invention may be embodied in other specific forms without
20 departing from its scope or essential characteristics. The described embodiments
21 are to be considered in all respects only as illustrative and not restrictive. The scope

1 of the invention is, therefore, indicated by the appended claims rather than by the
2 foregoing description. All changes which come within the meaning and range of
3 equivalency of the claims are to be embraced within their scope.

4 What is claimed is:

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21